Sergio Baretto - sergiob@imagecampus.edu.ar

Comparativa: C++ y Lua

Referencia rápida para comparar la sintaxis de los dos lenguajes, tomando sus elementos básicos.

No es una referencia ni una descripción completa de los lenguajes ni de sus características. Solo ejemplos para distinguir la sintaxis. Tampoco es una referencia exhaustiva sobre los elementos que se mencionan, sino sobre aspectos vistos en clase.

Para ampliar la información o indagar en otros recursos:

- https://devhints.io/lua
- https://www.lua.org/manual/

Sergio Baretto - sergiob@imagecampus.edu.ar

Programa básico

C++	Lua

`#include <iostream>`
 	
`void main()`
`{`
`    std::cout << "Hola";`
`}`
	`print("Hola")`
	
	IMPORTANTE: no hace falta poner punto y coma ; al final de una instrucción

Comentarios

C++	Lua

`// una linea`
 	
`/* varias`
`lineas  */`
	`-- una linea`
	
	`--[[ varias`
	`lineas ]]`

Sergio Baretto - sergiob@imagecampus.edu.ar

Tipos de dato fundamentales, Variables y Operadores

Lua es un lenguaje con un sistema de tipos de dato dinámico, lo cual es diferente a **C++**.

En concreto esto implica que no hace falta declarar el tipo de dato para una variable y que se le puede asignar un valor de un tipo distinto a una variable existente.

Más info en: https://www.lua.org/manual/5.4/manual.html#2.1

C++	Lua

`bool value = true;`
`int number = 10;`
`float anotherNumber = 10.5f;`
`std::string text = "hello world";`
	`value = true`
	`number = 10`
	`anotherNumber = 10.5`
	`text = "hello world"`
	
	`--[[ puedo asignar un dato de`
	`otro tipo a una variable`
	`que ya guardaba algo ]]`
	
	`value = "antes era bool"`

Las variables por default son globales a menos que se anteponga la keyword local a su declaración, lo cual es conveniente para limitar su scope al bloque de código, función o archivo que la contiene.

Más info en: http://www.lua.org/pil/4.2.html

En **Lua** los operadores son similares a **C++**, aunque revisar que en algunos casos cambian símbolos o se suman algunos. Ver por ejemplo:

- https://www.tutorialspoint.com/lua/lua_operators.htm

Sergio Baretto - sergiob@imagecampus.edu.ar

Input / Output

C++	Lua
`std::cout << "Hola" << std::endl;`	`print("Hola")` `io.write("Hola", "\n")`
`std::cout << "Hola";`	`io.write("Hola")`
`int edad = 10;` `std::cout << "Su edad es " << edad;`	`edad = 10;` `print("Su edad es " .. edad)`

`int num = 0;`
`std::cout << "Ingrese numero:";`
`std::cin >> num;`
	`io.write("Ingrese numero:")`
	`name = io.read()`

Constantes y Enumeradores

Lua no tiene **`const`** y **`enum`** de la misma forma que **C++**.

Hay alternativas y en versiones más actuales de **Lua** se incorporan constantes. Ver por ejemplo:

- https://www.lua.org/manual/5.4/manual.html#3.3.7
- https://unendli.ch/posts/2016-07-22-enumerations-in-lua.htm

Sergio Baretto - sergiob@imagecampus.edu.ar

Selección

if

C++	Lua

`if (x < y)`
`{`
`  // x es menor a y`
`}`
	`if (x < y) then`
	`  -- x es menor a y`
	`end`

`if (x < y)`
`{`
`  // x es menor a y`
`}`
`else`
`{`
`  // x NO es menor a y`
`}`
	`if (x < y) then`
	`  -- x es menor a y`
	`else`
	`  -- x NO es menor a y`
	`end`

`if (x < y)`
`{`
`  // x es menor a y`
`}`
`else if (x > y)`
`{`
`  // y es menor a x`
`}`
`else`
`{`
`  // son iguales`
`}`
	`if (x < y) then`
	`  -- x es menor a y`
	`elseif (x > y) then`
	`  -- y es menor a x`
	`else`
	`  -- son iguales`
	`end`
	
	IMPORTANTE: no se usan llaves {}, el **then**/**else** y el **end** funcionan como llaves para permitir varias instrucciones

switch

Lua no tiene una sentencia **`switch`** como sí tiene **C++**

Sergio Baretto - sergiob@imagecampus.edu.ar

Loops

while

C++	Lua

`while (i < 10)`
`{`
`  i++;`
`}`
	`while (i < 10) do`
	`  i = i + 1`
	`end`

do while

C++	Lua

`do`
`{`
`  i++;`
`}`
`while (i < 10);`
	`repeat`
	`  i = i + 1`
	`until (i == 10)`
	
	IMPORTANTE: No es exactamente el mismo comportamiento porque no es "repetir mientras" sino "repetir hasta".

for

C++	Lua

`for (int i = 0; i < 10; i++)`
`{`
`  // repite 10 veces (0 a 9)`
`}`
	`for i = 0, 9, 1 do`
	`  -- repite 10 veces (0 a 9)`
	`end`
	
	`--[[ alternativa: cuando el incremento es 1 se puede obviar ]]`
	
	`for i = 0, 9 do`
	`  -- repite 10 veces (0 a 9)`
	`end`

Sergio Baretto - sergiob@imagecampus.edu.ar

Random

C++	Lua

`// seed`
`// recordar agregar #include <ctime>`
`srand( time(0) );`
	`// seed`
	
	`math.randomseed( os.time() )`

`// un entero positivo aleatorio` `int aleatorio = rand();`	`-- un numero con coma entre 0 y 1 inclusive` `aleatorio = math.random()`
`// un int aleatorio entre 0 y 9` `int aleatorio = rand() % 10;`	`-- un entero aleatorio entre 0 y 9 inclusive` `aleatorio = math.random(0, 9)`
`// un int aleatorio entre 1 y 10` `int aleatorio = rand() % 10 + 1;`	`-- un entero aleatorio entre 1 y 10 inclusive` `aleatorio = math.random(10)`

Sergio Baretto - sergiob@imagecampus.edu.ar

Structs

C++	Lua

`struct Ball {`
`  int posX;`
`  int posY;`
`  float speed;`
`};`
 	
`Ball ball;`
 	
`ball.posX = 100;`
 	
`std::cout << ball.speed;`
	`--[[ No es exactamente lo mismo,`
	`  pero puede replicarse de`
	`  esta manera ]]]`
	
	`ball = {`
	`  posX = 10,`
	`  posY = 20,`
	`  speed = 12.5`
	`}`
	
	`ball.posX = 100`
	
	`print(ball.speed)`

Sergio Baretto - sergiob@imagecampus.edu.ar

Arrays

Inicialización

C++	Lua

`int nums[] = {1, 2, 3};`
 	
`int nums[3] = {1, 2, 3};`
 	
`int nums[3] = {}; // todos 0`
	`nums = {1, 2, 3}`
	
	`nums = {} --[[ array vacío, se le pueden asignar valores luego ]]`

Asignación

C++	Lua
`nums[0] = 10;`	`nums[1] = 10` **IMPORTANTE:** los índices comienzan en 1, no en 0

Lectura

C++	Lua
`int valor = nums[0];`	`valor = nums[1]` **IMPORTANTE:** los índices comienzan en 1, no en 0

Recorrido

C++	Lua

`const int cantTotal = 3;`
 	
`for (int i=0; i < cantTotal; i++)`
`{`
`  std::cout << nums[i];`
`}`
	`--[[ el numeral # seguido del nombre del array da el tamaño del mismo ]]`
	
	`for i = 1, #nums do`
	`  print(nums[i])`
	`end`
	
	IMPORTANTE: los índices comienzan en 1, no en 0

Sergio Baretto - sergiob@imagecampus.edu.ar

Funciones

Creación

C++	Lua

`void mostrarSaludo()`
`{`
`    std::cout << "Hola";`
`}`
 	
`void mostrarSuma(int x, int y)`
`{`
`    std::cout << x + y;`
`}`
 	
`int sumar(int x, int y)`
`{`
`    return x + y;`
`}`
	`function mostrarSaludo()`
	`    print("Hola")`
	`end`
	
	`function mostrarSuma(x, y)`
	`    print(x + y)`
	`end`
	
	`function sumar(x, y)`
	`    return x + y`
	`end`

Llamada

C++	Lua

`mostrarSaludo();`
 	
`mostrarSuma(2, 3);`
 	
`int resultado = sumar(2, 3);`
	`mostrarSaludo()`
	
	`mostrarSuma(2, 3)`
	
	`resultado = sumar(2, 3)`